**“****Resource Allocation in Disasters using Dynamic Programming”**

***A***

***Project Report***

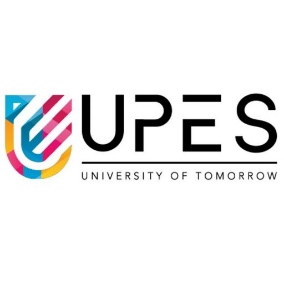
*submitted in partial fulfillment of the*

*requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE & ENGINEERING**



**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, Uttarakhand**

**November – 2024**

**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide **Ms. Silky Goel**, for all advice, encouragement and constant support she has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thanks to our respected **Dr. Anil Kumar,** **Head Department of SOCS,** for his great support in doing our project in **Resource Allocation in Disasters using Dynamic Programming.**

We are also grateful to Dean SoCS UPES for giving us the necessary facilities to carry out our project work successfully. We also thanks to our Activity Coordinator, Dr. Suneet Gupta for providing timely support and information during the completion of this project. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

**ABSTRACT**

The effective distribution of resources during disasters is critical to minimizing human suffering and ensuring timely relief efforts. Conventional approaches often struggle to address the complexities of disaster scenarios, where resource requirements are dynamic, and logistical constraints can impede swift action. This project, **"Resource Allocation in Disasters using Dynamic Programming,"** introduces a novel computational system designed to optimize and streamline the allocation process for essential supplies, including food, water, and medical aid.

The system utilizes **Dynamic Programming** to manage resource allocation efficiently, prioritizing regions based on severity while balancing inventory availability. Additionally, **Dijkstra’s shortest path algorithm** is implemented to calculate optimal transportation routes, ensuring rapid delivery with minimal delays. Cities and headquarters are represented as nodes in a graph, with distances or transportation costs assigned as weights to the connecting edges, enabling effective modeling of the disaster network.

An important feature of the system is its use of **Dear ImGui** for an intuitive graphical user interface, combined with **OpenGL-based visualization** to render the disaster network in real time. The interface enables users to input data interactively, observe network structures, analyze resource distribution paths, and dynamically adjust to changing conditions. This user-focused design enhances decision-making during critical moments.

The project offers a scalable, flexible solution for modern disaster management, addressing key challenges such as limited resources, changing priorities, and logistical barriers. By integrating computational optimization with real-time visualization and interactivity, this system provides a practical tool for improving the effectiveness of disaster response operations.

**TABLE OF CONTENTS**

**S.No. Contents Page No**

1. **Introduction 1**
   1. History 2
   2. Requirement Analysis 3
   3. Main Objective 4
   4. Sub Objectives 4
   5. Pert Chart Legend 5
2. **System Analysis 6**
3. **Tools and Technologies**
4. **Design 10**
5. **Implementation 34**
6. **Output screens 44**
7. **Limitations and Future Enhancements** 68
8. **Conclusion 69**

1. **INTRODUCTION**

Disasters, whether stemming from natural events such as earthquakes, floods, and hurricanes, or human-induced causes such as industrial accidents and armed conflicts, cause significant disruption to lives, infrastructure, and economies. The aftermath of these events necessitates the rapid deployment of resources to alleviate human suffering and prevent further loss of life. Ensuring the prompt and equitable distribution of essential resources such as food, water, and medical supplies presents a critical challenge in these high-stress scenarios.

Traditional approaches to disaster resource management often rely on manual processes and heuristic-based decision-making. While effective in certain contexts, these methods lack the speed, precision, and scalability required to address the complexities of large-scale disaster scenarios. The increasing severity and unpredictability of modern disasters, coupled with limited resource availability, highlight the need for systematic, technology-driven approaches to optimize resource allocation.

The project, **"Resource Allocation in Disasters using Dynamic Programming,"** seeks to bridge this gap by introducing a computational framework to manage and optimize the distribution of critical resources. By employing optimization algorithms and graph theory, the system models the network of affected cities and resource hubs, focusing on prioritizing areas based on urgency and minimizing delays and resource wastage. This framework ensures equitable distribution and enhanced efficiency in disaster response operations.

A key innovation of this project is the integration of advanced visualization tools using **Dear ImGui** and **OpenGL**, which provide an interactive graphical representation of the disaster network. These tools facilitate real-time monitoring of resource flows, offering decision-makers an intuitive interface to evaluate, adjust, and respond dynamically to evolving conditions. The transparency and accountability provided by these visualization features make the system not only efficient but also highly user-centric.

With the frequency and intensity of disasters increasing due to factors such as climate change and urbanization, there is an urgent need for modern, data-driven solutions. This project contributes to the advancement of disaster response mechanisms by leveraging technology to enable rapid, efficient, and equitable distribution of resources. The system’s integration of computational efficiency, dynamic prioritization, and real-time visualization positions it as a valuable tool in addressing the challenges of contemporary disaster management.

This report presents the conceptual framework, design, and implementation of the proposed system, marking a significant step toward smarter and more impactful disaster resource management solutions.

**Main Objective**

The overarching aim of this project is to design and implement a robust system for disaster resource allocation that leverages computational efficiency and visualization to enhance disaster response. Specifically, the system seeks to:

1. **Employ Dynamic Programming** to optimize the distribution of essential resources such as food, water, and medical supplies to regions affected by disasters.
2. **Facilitate efficient and equitable allocation** by prioritizing affected areas based on severity and proximity, thereby reducing response times and ensuring fairness in resource distribution.
3. **Integrate Dijkstra’s shortest path algorithm** to calculate optimal transportation routes between headquarters and disaster-hit cities, minimizing delays and preventing resource wastage.
4. **Utilize an intuitive visualization framework** based on OpenGL and Dear ImGui, enabling decision-makers to interact with a graphical representation of cities, routes, and headquarters, thus improving situational awareness and decision-making.
5. **Support real-time adaptability** by dynamically updating resource allocation and routing plans to address evolving disaster scenarios effectively.

**Sub Objectives**

1. **Graph Modeling**:

* Represent disaster-affected regions and resource hubs (headquarters) as nodes in a weighted graph.
* Model transportation routes with weighted edges, where weights correspond to distances or transportation costs, enabling efficient pathfinding and resource distribution.

2. **Resource Prioritization**:

* Categorize cities based on urgency, using predefined priority levels and resource availability as key criteria.
* Ensure equitable resource distribution among cities with similar priority levels, maintaining fairness in the allocation process.

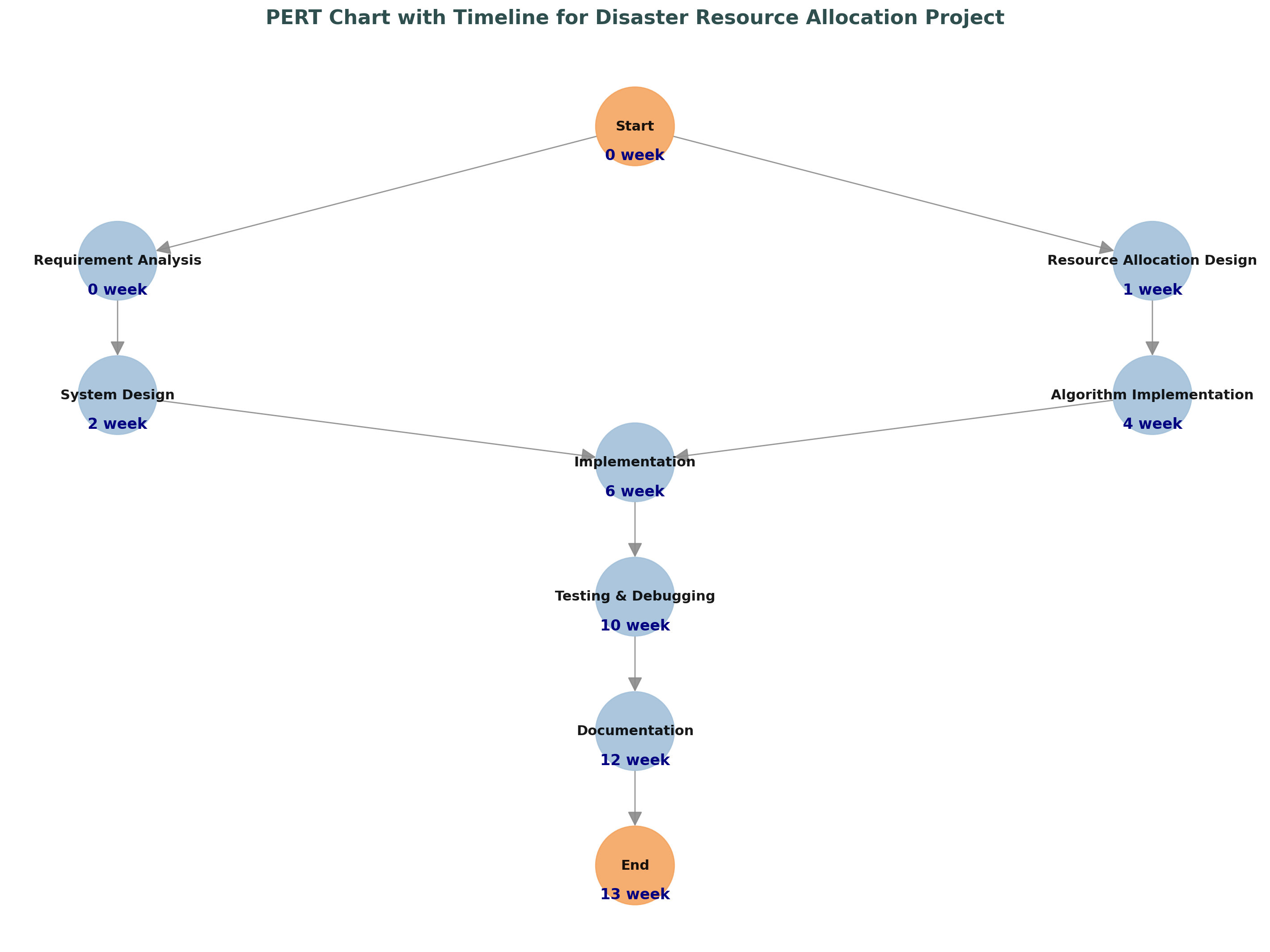
3. **Algorithm Implementation**:

* Implement **Dijkstra’s Algorithm** to compute the shortest paths between headquarters and cities, optimizing transportation routes.
* Use **Dynamic Programming** to allocate resources efficiently, minimizing wastage while meeting the needs of high-priority areas.

4. **Visualization**:

* Develop an interactive graphical interface using OpenGL to visualize the disaster network, including cities, routes, and resource flows.
* Display dynamic resource levels at each city and headquarters, providing decision-makers with real-time situational awareness.

**Pert Chart**



1. **SYSTEM ANALYSIS**

**2.1 Existing System**

Traditional approaches to disaster resource management predominantly rely on manual coordination, local resource stockpiling, and basic allocation strategies. While these methods can be effective in small-scale or predictable disaster scenarios, they often fall short when addressing the complexities of large-scale, dynamic crises. Several key characteristics and limitations of existing systems include:

1. **Manual Decision-Making**:
   * Resource allocation decisions are frequently guided by intuition or subjective judgment rather than data-driven methodologies, leading to inefficiencies and delays in delivering aid.
   * The absence of dynamic prioritization mechanisms often results in inequitable distribution, leaving some regions underserved while others may receive excessive aid.
2. **Static Route Planning**:
   * Current systems typically rely on predefined transportation routes that cannot adapt to disruptions such as damaged infrastructure or blocked roads.
   * These fixed logistics strategies often result in increased transportation costs and prolonged delivery times, further delaying aid to critical regions.
3. **Limited Data Utilization**:
   * Existing systems lack integration with real-time data sources such as IoT devices, satellite imagery, or sensor networks, which could provide valuable updates on disaster conditions.
   * Decision-making processes are often based on static, outdated data, rendering them ineffective in rapidly changing disaster scenarios where conditions evolve continuously.
4. **Lack of Visualization**:
   * Resource management tools rarely include visual aids to track resource movement, assess critical areas, or display the logistical network.
   * This lack of visualization impedes decision-makers from identifying bottlenecks, optimizing routes, or understanding the overall progress of relief efforts.

These limitations highlight the need for a more dynamic, data-driven, and adaptive approach to disaster resource management, integrating computational methods, real-time data, and interactive visualization to overcome the inefficiencies of traditional systems.

**Example Limitations**:

* During the 2004 Indian Ocean Tsunami, delays in aid distribution highlighted the inability to coordinate resources efficiently.
* The 2020 COVID-19 pandemic exposed vulnerabilities in global supply chains, demonstrating the need for adaptive and efficient disaster management systems.

**2.2 Motivations**

The limitations of traditional disaster management systems, combined with the increasing frequency and intensity of disasters, underscore the necessity for a scalable, adaptive, and efficient approach to resource allocation. The following key factors drive the development of this project:

1. **Growing Complexity of Disasters**:
   * The intensification of natural disasters due to climate change has resulted in greater unpredictability and larger-scale crises.
   * Rapid urbanization has led to densely populated areas in disaster-prone regions, significantly amplifying the demand for efficient resource allocation and timely response.
2. **Dynamic and Unpredictable Needs**:
   * The requirements for essential resources such as food, water, and medicine often change rapidly in disaster scenarios, influenced by evolving conditions and unexpected challenges.
   * Existing systems lack the ability to adapt dynamically to these changing demands, leading to inefficiencies and delays.
3. **Advancements in Computational Tools**:
   * Algorithms such as **Dijkstra’s shortest path** and **Dynamic Programming** provide powerful methods to optimize transportation routes and resource allocation, ensuring efficiency and minimizing waste.
   * Visualization frameworks, including **OpenGL** and **Dear ImGui**, enable intuitive monitoring and management of disaster networks, offering decision-makers a clear view of resource flows and network dynamics.
4. **Failures in Historical Disasters**:
   * The 2020 COVID-19 pandemic exposed significant flaws in global resource management systems, including delays, mismanagement, and insufficient transparency.
   * Inefficiencies observed during past disasters highlight the need for computationally driven solutions to prevent similar failures in future scenarios.
5. **Real-Time Data and IoT Integration**:
   * Advances in IoT, satellite imagery, and mobile technologies provide an opportunity to collect real-time data about disaster conditions, affected populations, and resource needs.
   * Utilizing this data allows for dynamic resource allocation, making systems more responsive to changing scenarios.

These motivations form the foundation for this project, emphasizing the importance of leveraging advanced computational tools and real-time data integration to address the critical challenges of disaster management effectively.

**2.3 Proposed System**

The proposed system adopts a computational framework to address the inefficiencies and limitations of traditional disaster resource management methods. By leveraging modern technologies, it ensures efficient and equitable distribution of resources during disaster scenarios. The system’s key features include:

1. **Dynamic Prioritization**:
   * Cities are ranked based on severity of impact and resource requirements, ensuring that the most critical areas receive aid first.
   * This prioritization mechanism dynamically adjusts to evolving disaster conditions, promoting equitable resource allocation.
2. **Graph-Based Model**:
   * The disaster network, comprising cities and headquarters, is modeled as a weighted graph.
   * Nodes represent locations (cities and headquarters), and edges represent transportation routes, with weights assigned based on distances or transportation costs.
3. **Optimization Algorithms**:
   * **Dynamic Programming** is employed to optimize resource allocation, minimizing wastage while ensuring timely delivery to high-priority regions.
   * **Dijkstra’s Algorithm** is used to compute the shortest transportation routes, enhancing efficiency and reducing delays in resource delivery.
4. **Real-Time Visualization**:
   * The system integrates **OpenGL and Dear ImGui based visualization** to provide an interactive and user-friendly graphical interface.
   * Decision-makers can monitor the network in real time, visualize resource flows, and make informed decisions dynamically.
5. **Scalability**:
   * The system is designed to handle a large number of cities and transportation routes efficiently.
   * Its architecture enables it to adapt dynamically to changes in disaster conditions, such as new routes or updated priorities.

This proposed system not only addresses the challenges of resource allocation in disaster management but also provides a strong foundation for future advancements. Potential enhancements include the integration of predictive analytics for proactive decision-making and IoT technologies for real-time data acquisition, further increasing the system’s adaptability and effectiveness.

1. **TOOLS AND TECHNOLOGIES**

The development of the **"Resource Allocation in Disasters using Dynamic Programming"** project incorporates various tools and technologies to achieve efficiency, scalability, and user interactivity. The key components are:

1. **Programming Language**:
   * **C/C++**:
     + Chosen for its high performance and low-level control, which are essential for implementing complex algorithms such as **Dynamic Programming** and **Dijkstra’s Algorithm**.
     + Its compatibility with graphics libraries like **OpenGL** ensures seamless integration of computational logic and visualization.
2. **Visualization**:
   * **OpenGL**:
     + A powerful, cross-platform graphics library used for rendering the real-time graphical interface.
     + Enables interactive monitoring by visualizing cities, headquarters, routes, and resource movements dynamically.
   * **Dear ImGui**:
     + A lightweight GUI framework integrated with OpenGL for intuitive user interactions.
     + Allows real-time input of data, visualization toggling, and results display, significantly enhancing the system's usability.
3. **Algorithms**:
   * **Dynamic Programming**:
     + Used to optimize resource allocation by dividing the problem into manageable subproblems.
     + Ensures minimal wastage and efficient distribution based on priority and proximity.
   * **Dijkstra’s Algorithm**:
     + Calculates shortest paths between nodes (headquarters and cities), optimizing transportation routes to reduce delays and transportation costs.
4. **Data Structures**:
   * **Graphs**:
     + Represent the disaster network with nodes as locations (cities and headquarters) and weighted edges as transportation routes.
     + Weighted edges capture costs or distances, enabling effective pathfinding and resource optimization.
   * **Adjacency Matrix**:
     + Stores graph data in a compact and efficient format for quick access and updates during calculations.
5. **Libraries**:
   * **Standard Template Library (STL)**:
     + Provides pre-built data structures like vectors, priority queues, and maps, simplifying the implementation of algorithms.
   * **OpenGL Utility Toolkit (GLUT)**:
     + Simplifies OpenGL programming for creating windows, rendering graphics, and handling user interactions.
   * **GLFW and GLAD**:
     + Frameworks used to manage OpenGL contexts and extend its functionality for improved rendering and visualization.

By combining computationally efficient algorithms, robust data structures, and intuitive visualization tools, this project delivers a comprehensive solution to the challenges of disaster resource allocation.

1. **DESIGN**

**Use Case Model for Requirement Analysis**

The **Administrator** serves as the primary actor in the system, responsible for overseeing disaster resource allocation. The system provides three core use cases to facilitate efficient management of resources, detailing the Administrator's interactions with the system.

**Actor and Use Cases**

1. **Actor**: Administrator
   * **Responsibilities**:
     + Add and manage cities affected by the disaster.
     + Allocate resources such as food, water, and medicine to high-priority regions.
     + Visualize the disaster network, including cities, routes, and real-time resource levels.

**Use Case Descriptions**

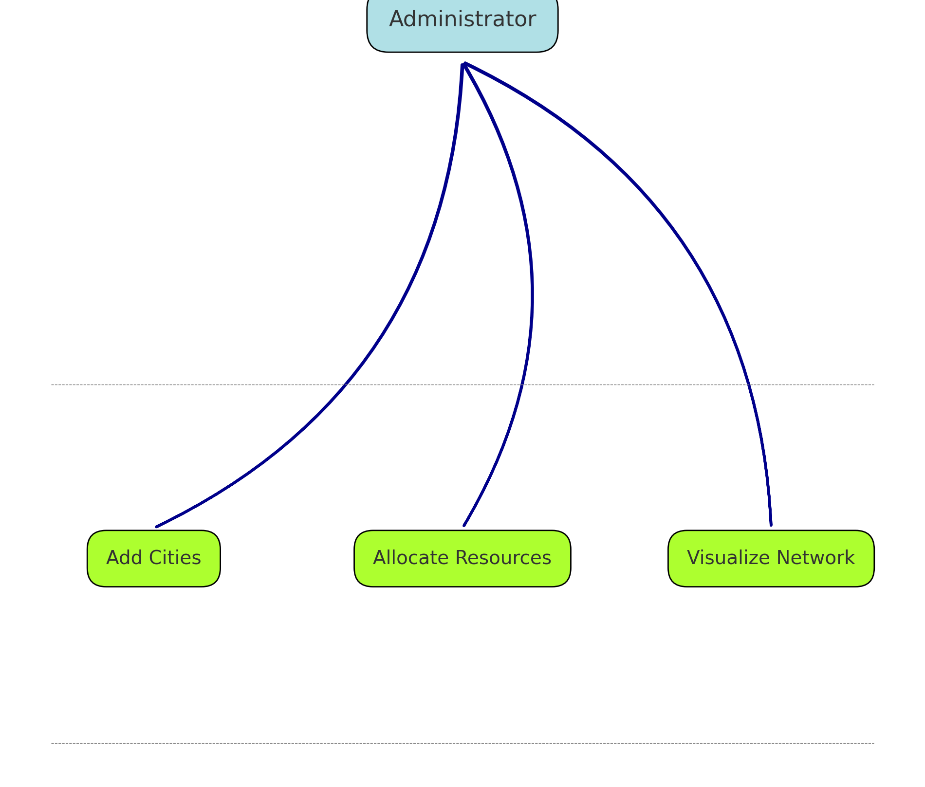
1. **Add Cities**:
   * **What the Actor Does**:
     + Inputs city details, including name, priority, and geographic coordinates.
     + Dynamically updates the disaster network by adding or removing cities as required.
   * **System Outcome**:
     + The city is integrated into the network and displayed on the real-time visualization interface.
2. **Allocate Resources**:
   * **What the Actor Does**:
     + Initiates resource allocation to cities based on priority and proximity to headquarters.
     + Monitors and validates resource availability at headquarters before allocation.
   * **System Outcome**:
     + Resources are efficiently distributed using **Dynamic Programming** for allocation and **Dijkstra’s Algorithm** for shortest path calculations, ensuring minimal delays and wastage.
3. **Visualize Network**:
   * **What the Actor Does**:
     + Monitors the disaster network to track resource flows and identify critical areas needing attention.
     + Adjusts decisions dynamically based on the real-time graphical representation.
   * **System Outcome**:
     + The network, including cities, routes, headquarters, and resource levels, is rendered interactively using **Dear ImGui** and **OpenGL**, enabling comprehensive situational awareness.

**Use Case Diagram Description**

In the **Use Case Diagram**:

* The **Administrator** is positioned at the center, interacting with the three primary use cases:
  1. Add Cities.
  2. Allocate Resources.
  3. Visualize Network.
* Each use case is depicted as a rectangle, connected to the Administrator through arrows, symbolizing the interaction flow.

The diagram visually encapsulates the system’s core functionalities, highlighting how the Administrator engages with different components to manage disaster resources efficiently.



**The Design Model**

The design model provides an architectural overview of the system, emphasizing modularity for scalability, maintainability, and efficiency. Key components include:

**Core Components**

1. **Cities and Headquarters**:
   * Represented as nodes in a graph.
   * **Cities**: Have attributes such as priority, coordinates, and resource requirements.
   * **Headquarters**: Serve as resource hubs with defined inventory levels for food, water, and medicine.
2. **Routes**:
   * Represented as edges in the graph, with weights indicating transportation costs or distances.
   * Dynamic updates to weights reflect real-time changes like roadblocks or delays.
3. **Resource Allocation Module**:
   * Uses **Dynamic Programming** to allocate resources efficiently based on priority and proximity.
   * Adapts dynamically to changing disaster needs, minimizing wastage.
4. **Shortest Path Module**:
   * Implements **Dijkstra’s Algorithm** to calculate optimal routes for resource transportation.
   * Ensures minimal delays and optimal resource delivery.
5. **Visualization Module**:
   * Renders the disaster network in real time using **OpenGL** and **Dear ImGui**.
   * Displays cities, routes, and resource levels, enabling interactive monitoring and decision-making.

**Object and Class Design**

The system is implemented procedurally while incorporating modular functions to ensure reusability and maintainability. Below are the key functional modules and their responsibilities, mapped to their respective features in the code:

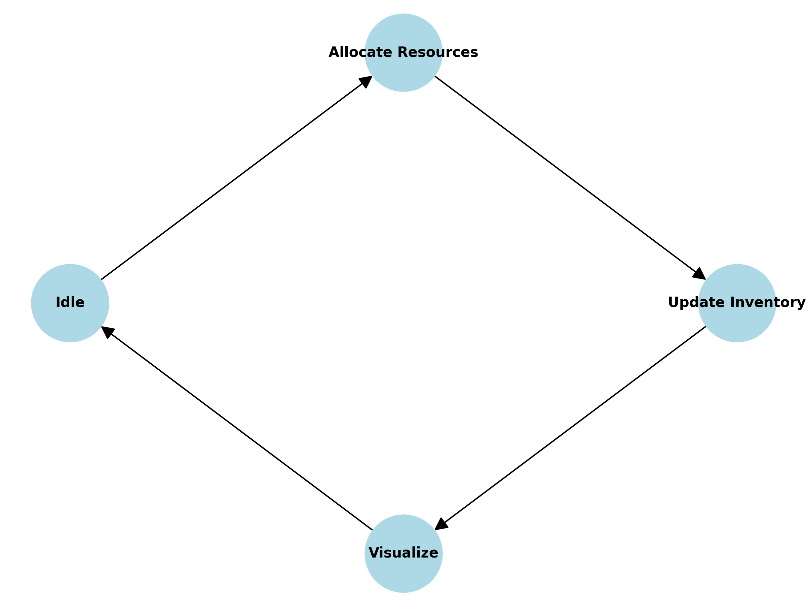
1. **City Module**:
   * **Attributes**:
     + points[MAX\_CITIES][2]: Stores the (x, y) coordinates of cities and headquarters for visualization.
     + resources[MAX\_CITIES][3]: Tracks the current levels of food, water, and medicine for each city and headquarters.
   * **Functions**:
     + **Dynamic Priority Management**: Resource prioritization is managed dynamically through functions like allocateResources.
     + **Request Resource Visualization**: Cities generate resource demands interactively through ImGui in functions like renderInputs and allocateResources.
2. **Resource Allocation Module**:
   * **Attributes**:
     + Resource inventories are stored in the resources array for headquarters.
   * **Functions**:
     + **allocateResources()**:
       - Finds the nearest headquarters for a given city using Dijkstra’s algorithm.
       - Allocates requested resources (medicine, food, water) while ensuring inventory constraints.
     + **updateResources()**:
       - Allows real-time updates to the resource levels at headquarters via ImGui inputs.
3. **Graph Module**:
   * **Attributes**:
     + graph[MAX\_CITIES][MAX\_CITIES]: Adjacency matrix representing the network of cities and routes, with weights indicating distances or transport costs.
   * **Functions**:
     + **renderInputs()**:
       - Enables users to input adjacency matrix values dynamically via ImGui.
     + **calculateShortestPath()**:
       - Implements Dijkstra’s algorithm to calculate the shortest path between headquarters and cities.
       - Provides real-time results, including the shortest path distance and route visualization.
4. **Visualization Module**:
   * **Attributes**:
     + Rendering is handled through **OpenGL** and **Dear ImGui**.
   * **Functions**:
     + **renderVisualization()**:
       - Draws cities, headquarters, routes, and resource flows dynamically.
       - Displays network elements as nodes and edges, using colors to differentiate cities and headquarters.
     + **displayAdjacencyMatrix()**:
       - Presents the adjacency matrix in a table format, ensuring transparency of graph connections.

**State Transition**

The system operates through the following key states, transitioning dynamically based on user inputs:

1. **Idle**:
   * The system waits for user actions, such as adding cities, allocating resources, or visualizing the network.
2. **Allocate Resources**:
   * Resources are distributed to cities using **Dijkstra’s Algorithm** for routing and **Dear ImGui** for user interaction, ensuring efficient and prioritized allocation.
3. **Update Inventory**:
   * Resource inventories at headquarters and cities are updated via the updateResources() function, reflecting changes after allocation.
4. **Visualize**:
   * The updated disaster network, including cities, routes, and resource levels, is rendered dynamically using **OpenGL** and **Dear ImGui**.

**State Transition Diagram**:



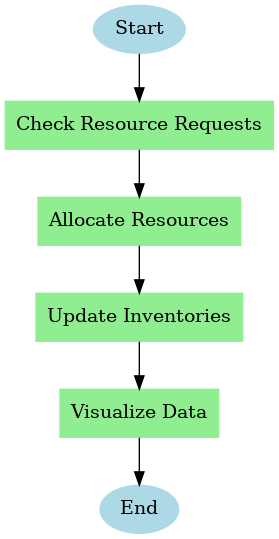
This cycle continues until all requests are addressed.

**Activity Diagram**

The activity diagram represents the logical flow of operations performed by the system, detailing the sequence of actions from initialization to the completion of resource requests. The process ensures efficient resource allocation and visualization.

**Steps:**

1. **Start**:
   * The system initializes and loads data for the network of cities and headquarters.
2. **Check Resource Requests**:
   * The system evaluates incoming requests for food, water, and medical supplies based on city priority levels.
3. **Allocate Resources**:
   * Resources are distributed dynamically using the allocation module, leveraging **Dijkstra’s Algorithm** and proximity analysis.
4. **Update Inventories**:
   * Resource levels at headquarters and cities are updated to reflect the changes after allocation.
5. **Visualize Data**:
   * The updated network is rendered using **OpenGL**, displaying cities, routes, and resource levels dynamically.
6. **End**:
   * The process completes when all pending requests have been fulfilled.

****

**5. Implementation**

The Disaster Resource Allocation System (DRAS) is designed to efficiently manage and allocate resources during emergencies. This section details the system’s implementation, covering real-world scenarios, core algorithms, and the data structures used.

**Scenarios**

The system addresses the following real-world scenarios:

1. **Adding City**:
   * New cities can be added dynamically, including attributes such as name, priority, coordinates, and resource levels.
   * Cities are visualized on a 2D map, showcasing their spatial relationships.
2. **Removing City**:
   * Cities and their associated routes can be removed, with the system dynamically updating the network.
3. **Resource Request Handling**:
   * Cities in need of resources generate requests specifying required quantities of food, water, and medicine.
   * The system prioritizes cities based on severity and allocates resources from the nearest capable HQ.
4. **Route Management**:
   * Routes between cities can be added or removed, with changes seamlessly reflected in the adjacency matrix.
5. **Visualization**:
   * The system uses OpenGL to provide a real-time graphical interface, representing cities, routes, and HQs for improved situational awareness.
6. **Resource Allocation**:
   * Resources are distributed based on city priority, shortest path to HQs, and inventory availability. The system ensures optimal delivery paths and equitable distribution.

**Algorithms**

1. **Shortest Path Calculation**:
   * **Purpose**: Finds the shortest route from HQs to cities, minimizing delivery time and cost.
   * **Implementation**: Uses **Dijkstra’s Algorithm** on the weighted graph representing the disaster network.
2. **Resource Allocation**:
   * **Purpose**: Allocates resources efficiently based on priority, distance, and availability.
   * **Workflow**:
     1. Identify the highest-priority city.
     2. Calculate the nearest HQ using the shortest path algorithm.
     3. Allocate resources and update inventories for both HQ and city.
3. **Dynamic Graph Management**:
   * **Purpose**: Updates the adjacency matrix whenever cities or routes are modified.
   * **Application**: Maintains the network’s accuracy for subsequent operations.

**Data Structures**

1. **Graph**:
   * The city network is stored as an adjacency matrix, where cells represent distances between cities. A value of 0 denotes no direct route.
2. **Resource**:
   * Resource levels for HQs and cities are maintained in a centralized structure, enabling efficient updates during allocation.
3. **City**:
   * Cities are stored with attributes such as name, priority, coordinates, and resource levels, ensuring accurate prioritization and visualization.

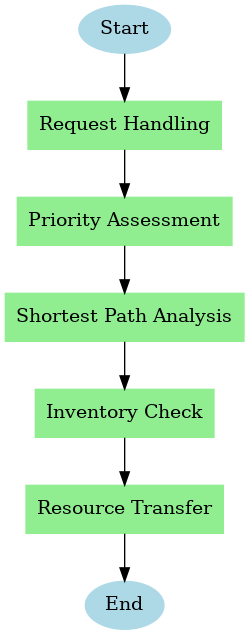
**Visualization Framework**

The system leverages OpenGL and Dear ImGui for real-time graphical representation:

1. **City Representation**:
   * Cities are displayed as polygons with distinct colors and labeled with their names and resource levels.
2. **HQ Representation**:
   * Headquarters are marked with red squares and labeled for easy identification.
3. **Route Visualization**:
   * Routes are drawn as lines connecting cities, with updates dynamically reflected in the visualization.
4. **Resource Indicators**:
   * Resource levels for each city are displayed adjacent to their graphical representation for quick monitoring.

**Resource Allocation Workflow**

1. **Request Handling**:
   * Cities submit requests specifying required resources.
2. **Priority Assessment**:
   * Cities are ranked based on their priority levels.
3. **Shortest Path Analysis**:
   * The system computes the shortest route from HQs to cities using Dijkstra’s Algorithm.
4. **Inventory Check**:
   * HQ inventories are evaluated to ensure resource availability.
5. **Resource Transfer**:
   * Resources are allocated from the nearest HQ, and inventories are updated in real time.

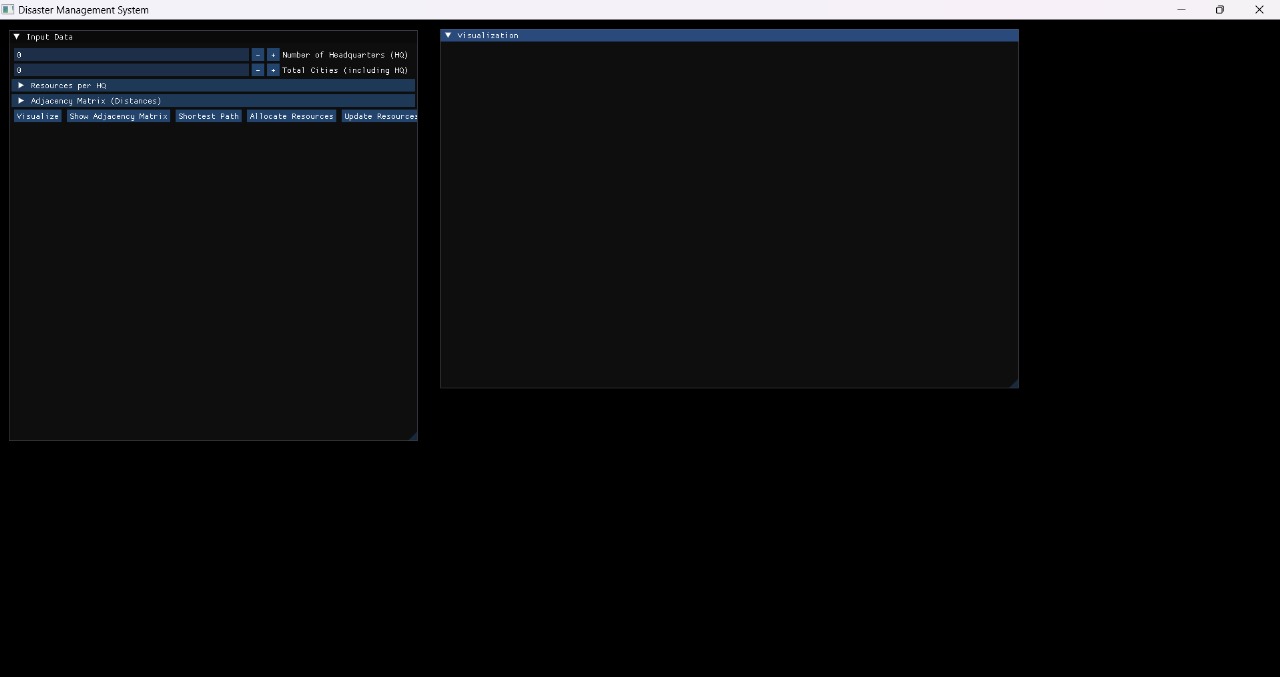


**System Architecture**

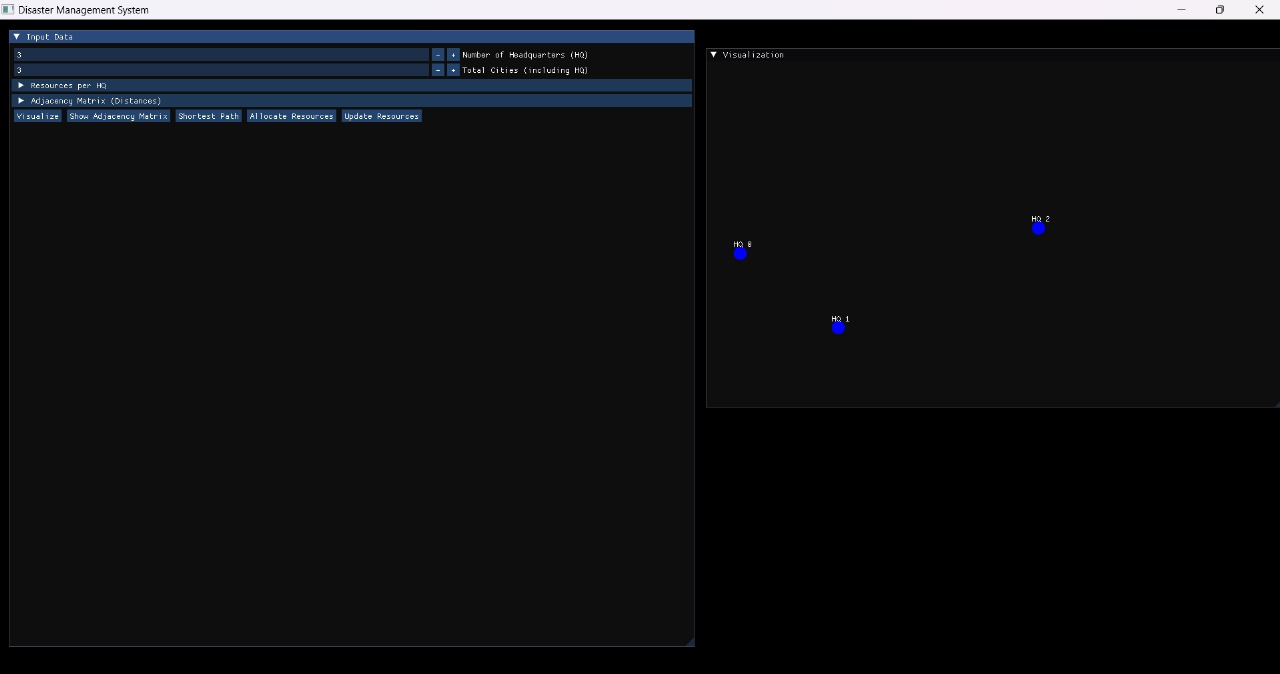
The architecture of the Disaster Resource Allocation System (DRAS) is designed to integrate its core functionalities seamlessly, ensuring efficiency, usability, and adaptability. The system is composed of the following primary components:

1. **Core Logic**:
   * Handles the management of cities, headquarters, and routes as a dynamic graph structure.
   * Implements key functionalities such as resource request handling, shortest path calculation using **Dijkstra’s Algorithm**, and dynamic resource allocation via **Dynamic Programming**.
   * Updates the adjacency matrix to reflect changes in the network, ensuring real-time accuracy.
2. **Graphical User Interface (GUI)**:
   * Built using **Dear ImGui** and **OpenGL**, the GUI provides a real-time, interactive interface for system visualization.
   * Displays cities, headquarters, routes, and resource levels dynamically, allowing users to interact with and monitor the system.
   * Facilitates actions such as adding/removing cities and routes, viewing the adjacency matrix, and visualizing resource allocations.
3. **Database Management**:
   * Resource inventories for cities and headquarters are stored in centralized arrays (resources[MAX\_CITIES][3]).
   * Tracks allocation history and logs updates, providing a complete view of resource distribution.
   * Ensures data consistency and enables efficient queries during resource allocation and network management.
4. **Output screens**

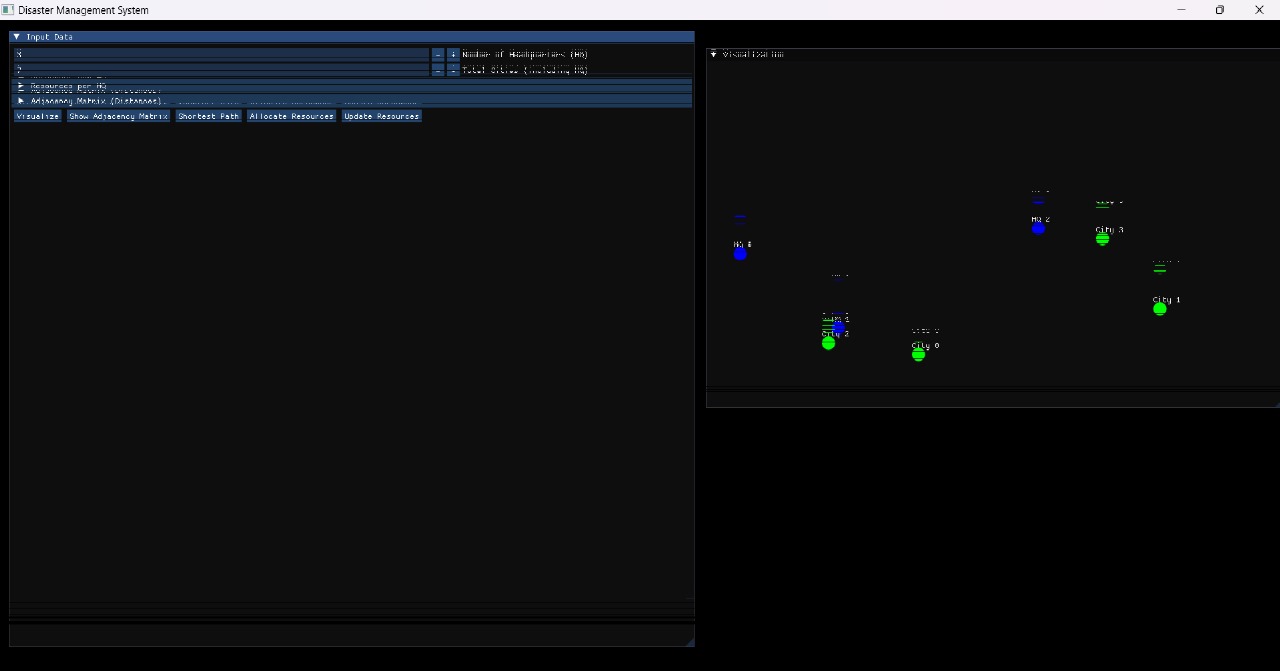
**1 - INPUT/OUTPUT WINDOW**



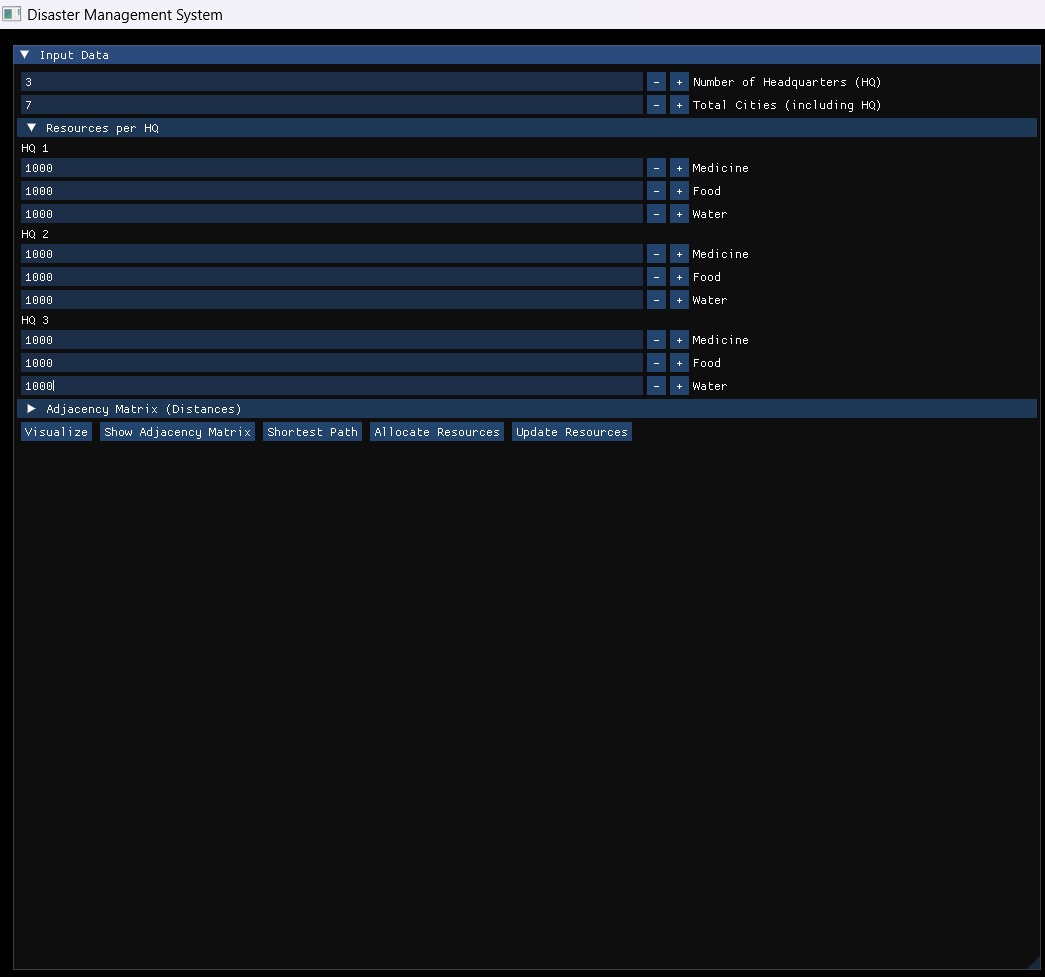
**2 - ADDING HEADQUARTERS**



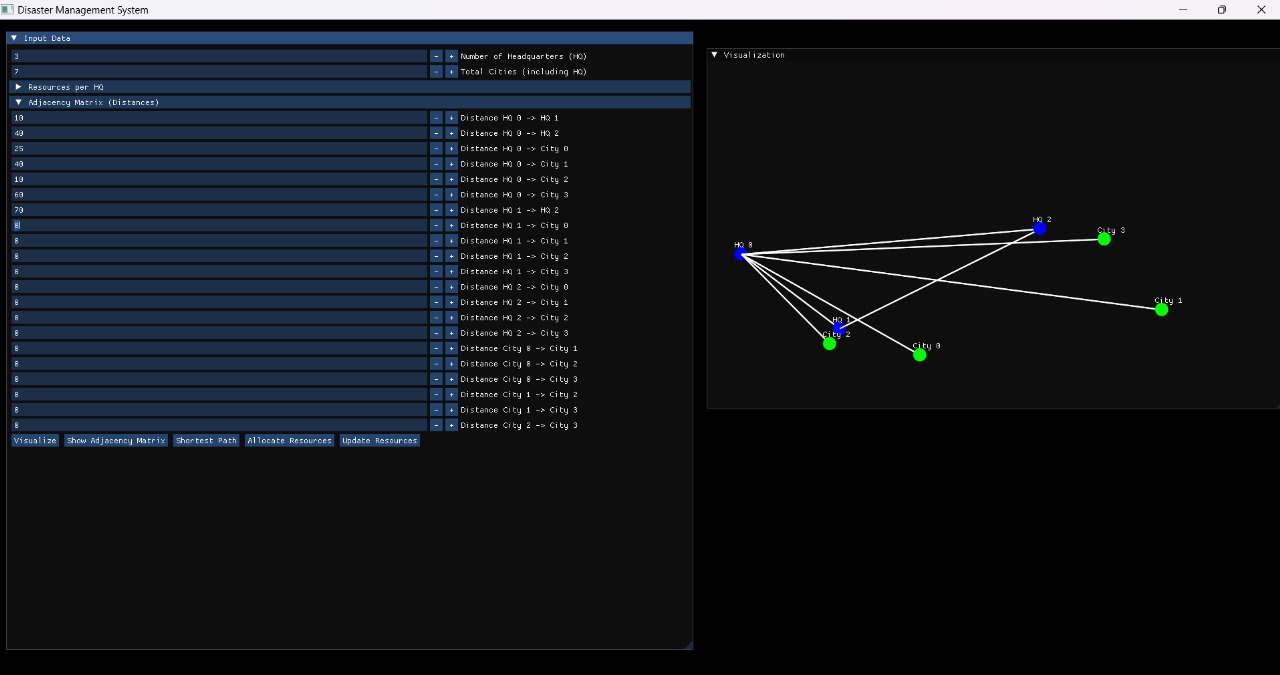
**3 – ADDING CITIES**



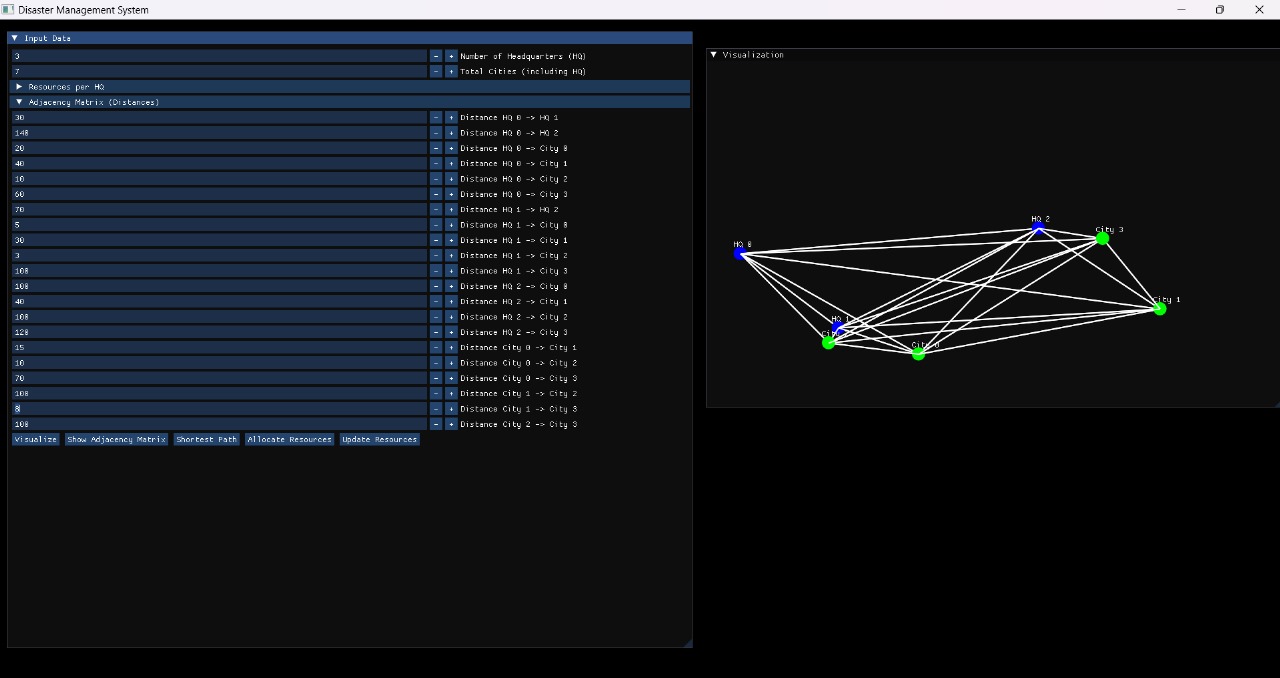
**4 - PROVIDING RESOURCES TO HEADQUARTERS**

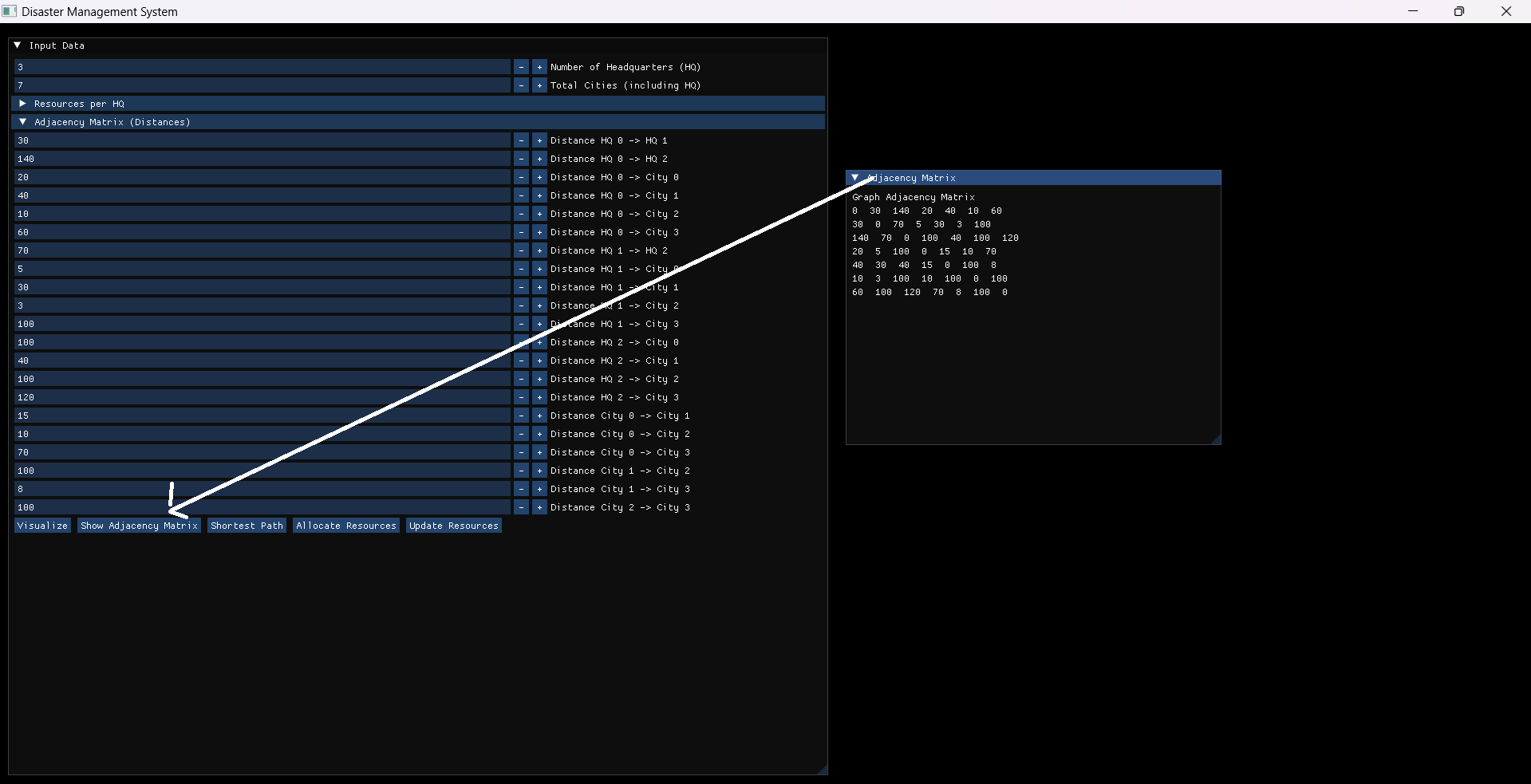


**5 - ADDING ROUTES BETWEEN CITIES ,HEADQUARTERS**

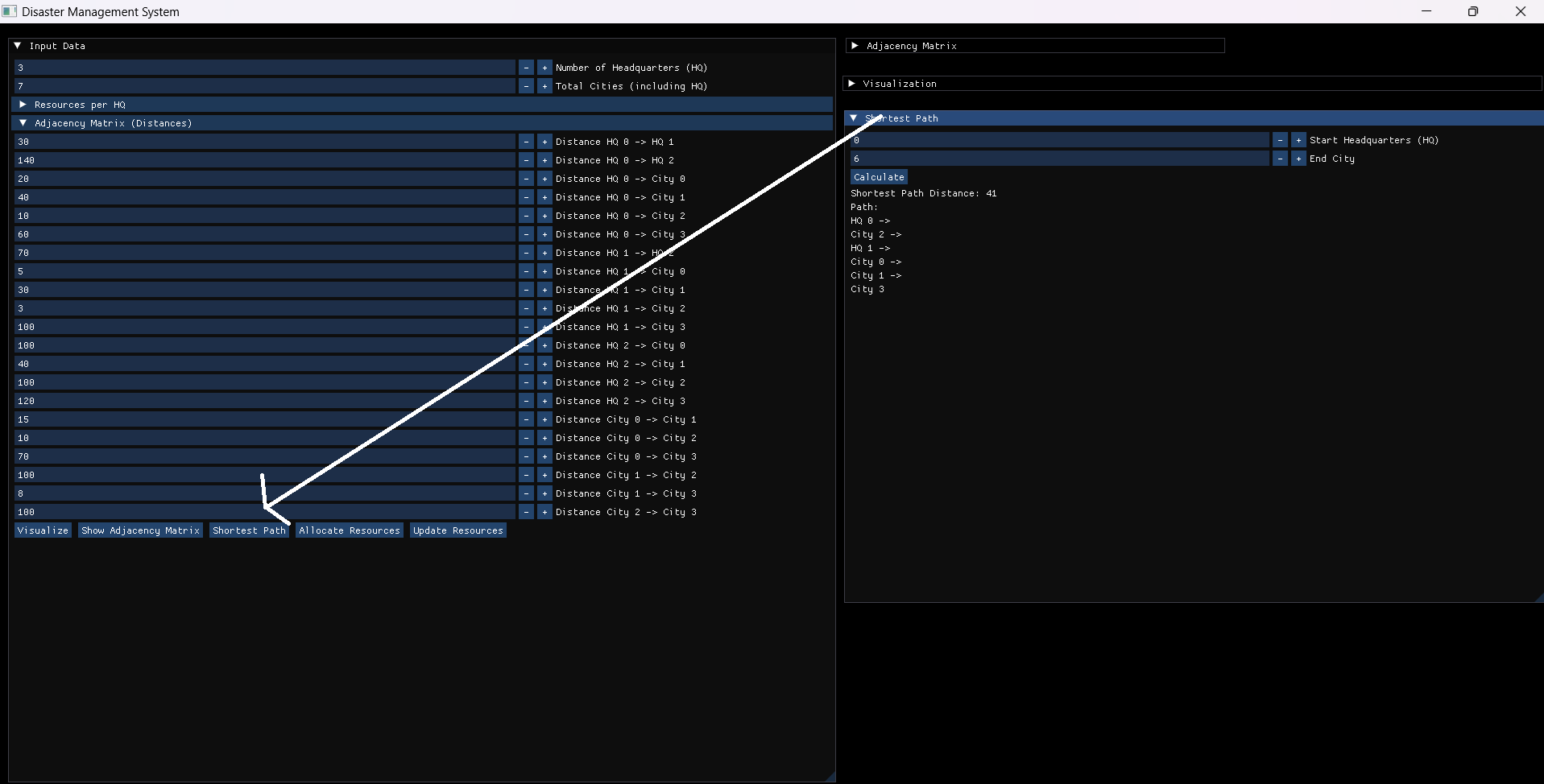


**6 - REAL TIME UPDATION OF ROUTES**

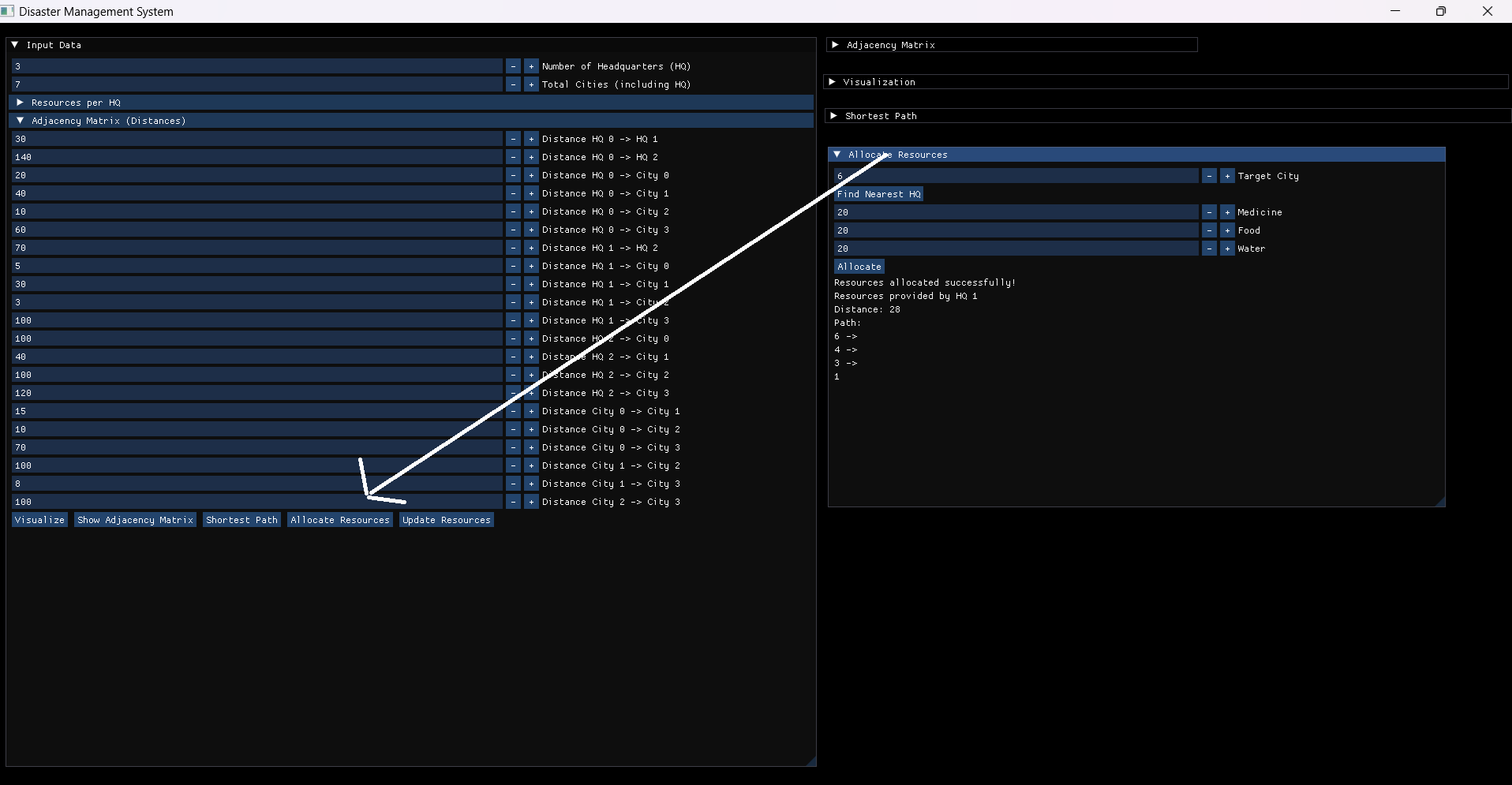


**7 - ADJACENCY MATRIX**

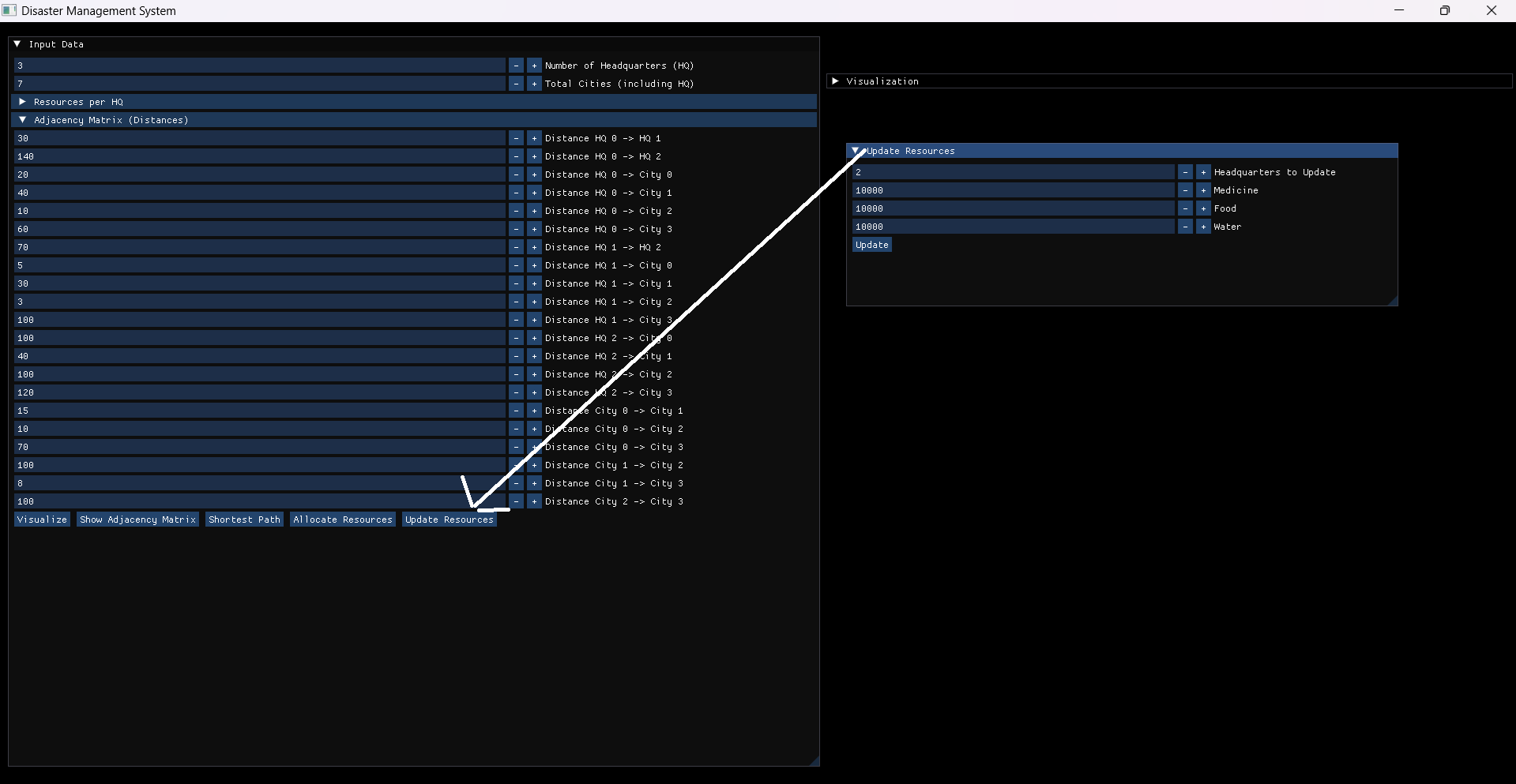
**8 - SHORTEST PATH USING DIJKSTRA ALGORITHM (HEADQUARTERS ARE REPRESENTED AS 0 ,1,2( FOR 3 HEADQUARTERS) AND CITIES AS 3,4,5,6 (FOR 4 CITIES))**



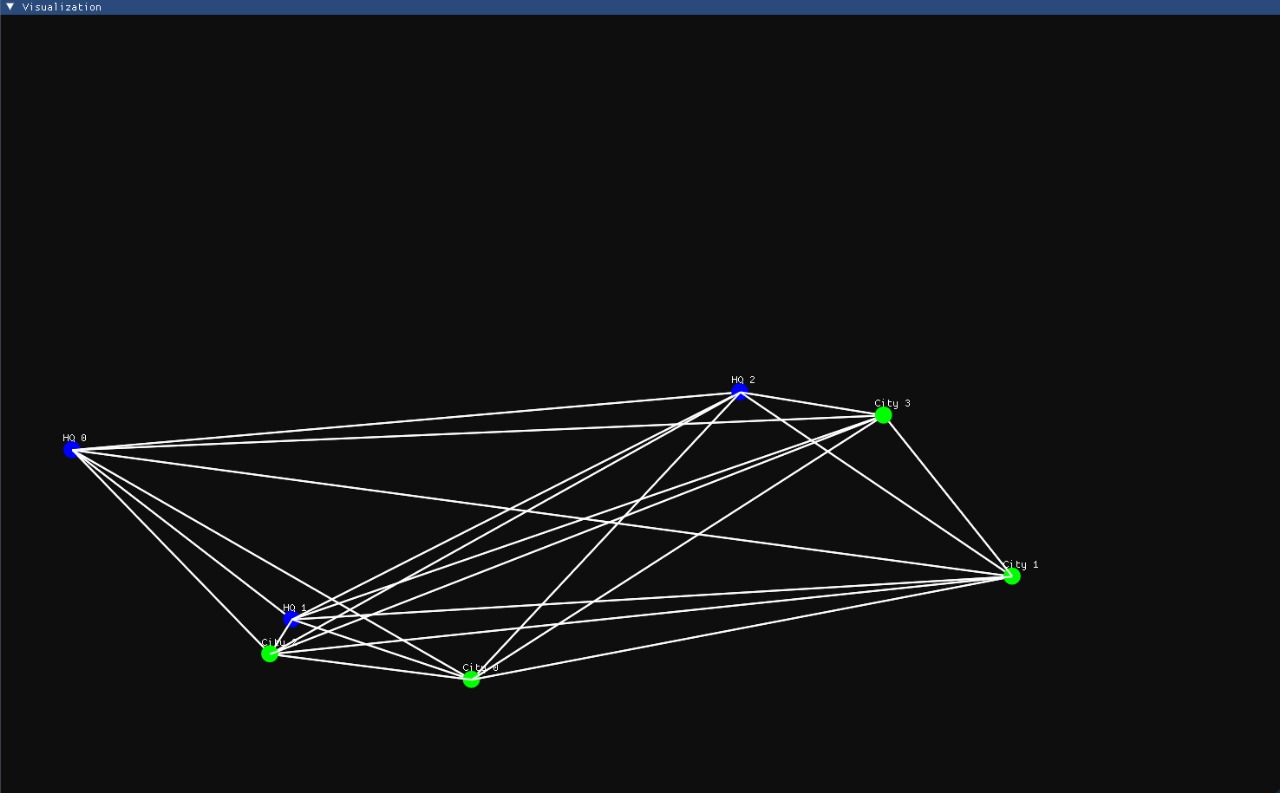
**9 - ALLOCATION OF RESOURCES FROM NEAREST HEADQUARTER**



**10 -** **UPDATION OF RESOURCES FOR HEADQUARTERS**

****

**11 – VISUALIZATION**



1. **Limitations and Future Enhancements**

**Limitations of the Disaster Resource Allocation System**

1. **Limited Visualization**:
   * The OpenGL-based visualization lacks interactive features like zooming, panning, or dynamic updates for deeper analysis.
2. **No Visualization for the Optimal Path:**
   * The Optimal Path is not highlighted by the system.
3. **Sequential Processing**:
   * Without multithreading, the system processes tasks sequentially, which may be inefficient for large-scale or simultaneous requests.
4. **Plain Text Logging**:
   * Logs are stored as plain text, lacking robust security and scalability for managing large data volumes.

**Future Enhancements**

1. **Advanced Allocation Strategies**:
   * Use AI models to account for factors such as transport time, demand urgency, and resource availability.
2. **Visualization of the Optimal Path:**
   * The optimal path could have been highlighted for better user experience.
3. **Predictive Resource Allocation**:
   * Utilize machine learning to predict future resource needs based on historical disaster data and trends.

By addressing these limitations and implementing the proposed enhancements, the system can evolve into a robust, scalable, and intelligent disaster management tool, better equipped to handle real-world emergencies efficiently.

1. **Conclusion**

The project titled **"Resource Allocation in Disasters using Dynamic Programming"** addresses one of the most critical challenges in disaster management: the efficient and equitable distribution of resources to disaster-affected regions. By leveraging optimization algorithms, graph-based modeling, and visualization techniques, the system ensures timely and effective resource allocation, ultimately reducing human suffering and preventing resource wastage.

The system integrates **Dynamic Programming** for resource optimization and **Dijkstra’s Algorithm** for shortest path calculations, enabling efficient transport of resources to high-priority areas. The use of OpenGL for real-time visualization provides decision-makers with a clear understanding of the disaster network, including cities, headquarters, routes, and resource flows. These features collectively make the system a powerful tool for modern disaster response.

This project has demonstrated several key achievements:

1. **Efficiency**: The system optimizes resource distribution, minimizing delays and transportation costs.
2. **Equity**: Resources are allocated based on the severity of needs, ensuring fairness among affected regions.
3. **Transparency**: The visualization module offers clear insights into resource movements and allocations.

Despite these accomplishments, the system has certain limitations, including fixed headquarters, reliance on static inputs, and limited real-time adaptability. These constraints highlight potential areas for future enhancement, such as integrating IoT for real-time data collection, machine learning for predictive modeling, and dynamic repositioning of headquarters.

In conclusion, this project provides a scalable, adaptable, and practical solution for disaster resource allocation. By combining computational efficiency with intuitive visualization, it sets a strong foundation for future advancements in disaster management systems. The system has significant potential to be expanded and integrated into real-world applications, contributing to more effective disaster relief efforts globally.

1. **References**

[1] Kumar, J. S., & Zaveri, M. A. (2017, November). Graph-based resource allocation for disaster management in IoT environment. In *Proceedings of the Second International Conference on Advanced Wireless Information, Data, and Communication Technologies* (pp. 1-6).[1]

Link : <https://dl.acm.org/doi/abs/10.1145/3231830.3231842>

[2] Sarma, D., Das, A., & Bera, U. K. (2020). An optimal redistribution plan considering aftermath disruption in disaster management. *Soft Computing*, *24*, 65-82.[2]

Link : <https://link.springer.com/article/10.1007/s00500-019-04287-7>

Wang, Y., Lyu, M., & Sun, B. (2024). Emergency resource allocation considering the heterogeneity of affected areas during the COVID-19 pandemic in China. *Humanities and Social Sciences Communications*, *11*(1), 1-15.[3]

Link : <https://www.nature.com/articles/s41599-024-02712-3>

Kumar, J. S., Zaveri, M. A., & Choksi, M. (2018). Activity based resource allocation in IoT for disaster management. In *Future Internet Technologies and Trends: First International Conference, ICFITT 2017, Surat, India, August 31-September 2, 2017, Proceedings 1* (pp. 215-224). Springer International Publishing.[4]

Link : <https://eudl.eu/doi/10.1007/978-3-319-73712-6_22>

Sarma, D., Das, A., Dutta, P., & Bera, U. K. (2020). A cost minimization resource allocation model for disaster relief operations with an information crowdsourcing-based MCDM approach. *IEEE transactions on engineering management*, *69*(5), 2454-2474.[5]

Link : <https://scholar.google.com.pk/citations?user=BPIxs5gAAAAJ&hl=th>

Dave, R. K. (2018). *Disaster management in India: Challenges and strategies*. Prowess Publishing.[6]

Link:[https://books.google.co.in/books?hl=en&lr=&id=QCpNDwAAQBAJ&oi=fnd&pg=PT7&dq=%5B6%5D+Dave,+R.+K.+(2018).+Disaster+management+in+India:+Challenges+and+strategies.+Prowess+Publishing.&ots=4bVfsRevz7&sig=FAtnGuXBr3n4A1WvDjIcmHmU3TM&redir\_esc=y#v=onepage&q=%5B6%5D%20Dave%2C%20R.%20K.%20(2018).%20Disaster%20management%20in%20India%3A%20Challenges%20and%20strategies.%20Prowess%20Publishing.&f=false](https://books.google.co.in/books?hl=en&lr=&id=QCpNDwAAQBAJ&oi=fnd&pg=PT7&dq=%5B6%5D+Dave,+R.+K.+(2018).+Disaster+management+in+India:+Challenges+and+strategies.+Prowess+Publishing.&ots=4bVfsRevz7&sig=FAtnGuXBr3n4A1WvDjIcmHmU3TM&redir_esc=y)

Gamitl, H., Kumar, J. S., & Zaveri, M. A. (2018, April). Resource monitoring and scheduling for post disaster management using internet of things. In *2018 3rd International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE[7]

Link : <https://ieeexplore.ieee.org/abstract/document/8529785>

Banerjee, S., & Mohapatra, S. (2024). Managing Disasters in a Compounding Scenario: A Case of Adaptive Governance in the State of Odisha, India. *International Journal of Climate Change: Impacts & Responses*, *16*(1).[8]

Link:<https://eds.p.ebscohost.com/abstract?site=eds&scope=site&jrnl=18357156&AN=178018557&h=aodM5CI67Tmi2lha2Zoiia2IPiX5OIUiJ%2fAhztQyRoIl4CZgoBLNX55bSdFYti1jWdvLKofqfYhllzLBLz3jWg%3d%3d&crl=c&resultLocal=ErrCrlNoResults&resultNs=Ehost&crlhashurl=login.aspx%3fdirect%3dtrue%26profile%3dehost%26scope%3dsite%26authtype%3dcrawler%26jrnl%3d18357156%26AN%3d178018)>